Trie cheatsheet for coding interviews

Introduction

Tries are special trees (prefix trees) that make searching and storing strings more efficient. Tries have many practical applications, such as conducting searches and providing autocomplete. It is helpful to know these common applications so that you can easily identify when a problem can be efficiently solved using a trie.
Be familiar with implementing from scratch, a `Trie` class and its `add`, `remove` and `search`methods.

Learning resources

- Readings
  - Trying to Understand Tries, basecs
  - Implement Trie (Prefix Tree), LeetCode
- Additional (only if you have time)
  - Compressing Radix Trees Without (Too Many) Tears, basecs

Time complexity

`m` is the length of the string used in the operation.

Operation	Big-O
Search	O(m)
Insert	O(m)
Remove	O(m)

Corner cases

- Searching for a string in an empty trie

- Inserting empty strings into a trie

Techniques

Sometimes preprocessing a dictionary of words (given in a list) into a trie, will improve the efficiency of searching for a word of length k, among n words. Searching becomes O(k) instead of O(n).

Essential questions

These are essential questions to practice if you're studying for this topic.

- Implement Trie (Prefix Tree)

Recommended practice questions

These are recommended questions to practice after you have studied for the topic and have practiced the essential questions.

- Add and Search Word
- Word Break
- Word Search II